INFORMATION AND CONTROL 67, 2–11 (1985)

Alternation with Restrictions on Looping

J. M. ROBSON

Department of Computer Science,
Australian National University,
Canberra, Australia

Alternating Turing machines with restrictions preventing them from returning to a previous configuration model games with rules enforcing such a restriction, for instance, the Chinese version of Go. Such restrictions do not affect the time complexity of problems for alternating Turing machines but space S on a machine with the restriction is equivalent either to time or to space exponential in S on a normal alternating machine, depending on the precise nature of the restriction.

1. INTRODUCTION

Alternating Turing machines (ATMs) and other forms of alternation have been considerably studied. They provide numerous interesting complexity results because their space and time complexity are related to deterministic time and space complexity in simple and well understood ways (Chandra, Kozen, and Stockmeyer, 1981). In particular they have been used to prove several results about the complexity of games (Fraenkel and Lichtenstein, 1981; Robson, 1983; Robson 1984a).

The work reported here was originally motivated by games with a rule that a player may not move so as to return the game configuration to one that has occurred previously. Some results about games with such a rule are known (Lichtenstein and Sipser, 1980; Robson, 1984b). This paper generalises these results by studying ATMs with restrictions which prevent them from returning to a previous configuration, that is, to a previous instantaneous description, either absolutely or conditionally. The time and space complexity of these new ATMs are studied and their relation to deterministic complexity established. First we define the new types of ATM to be discussed.

DEFINITION. A non-looping alternating Turing machine (NLATM) is an ATM which cannot return to a previous configuration. If, because of this restriction, a computation arrives at a configuration from which no next step is possible, the computation is regarded as having accepted its input iff the final state is universal.

0019-9958/85 $3.00

DEFINITION. A restricted looping alternating Turing machine (RLATM) is an ATM with two of its states designated as "clearing states"; the RLATM can return to a previous configuration, if and only if, it has entered both of the clearing states at least once since the same configuration occurred last. Acceptance is defined as for the NLATM.

Time and space complexities for NLATMs and RLATMs are defined in the usual ways. We will write NLATM f space for the set of languages accepted by a NLATM in space $O(f)$, NLATM Pspace for the union of NLATM f space over all polynomials f, NLATM exp space for the union of NLATM 2^f space over all polynomials f (which some writers more strictly would call NLATM polyexp space) and NLATM double exp space for the union of NLATM 2^{2^f} space over all polynomials f. Similar terminology will be used for time and for complexity classes over RLATMs, ATMs and (deterministic) TMs.

Note that the definition of the outcome of a computation which reaches a "dead end" is the natural extension of the normal definition that an existential configuration is accepting if there is some possible next step which produces an accepting configuration but a universal configuration is accepting if all possible next steps produce accepting configurations.

The flavour of the results to be proved can be indicated by giving here some simple corollaries (in each case the last equality is already well known and the others are proved here):

RLATM Ptime = NLATM Ptime = ATM Ptime (= TM Pspace).

NLATM Pspace = ATM exp time (= TM exp space).

RLATM Pspace = ATM exp space (= TM double exp time).

Section 2.1 will establish some fairly obvious results including the time complexity of $N(R)$LATMs. Sections 2.2 and 2.3 will establish theorems showing that TM exp space = NLATM Pspace and that TM double exp time = RLATM Pspace; the first of these equalities closely parallels the well-known result that a game in NLATM log space is complete in (TM) Pspace (Lichtenstein and Sipser, 1980) but the second appears to have no known analogue.

We find it clearest to discuss ATMs in terms of a game between players E and U with E choosing the action in existential states and U in universal states, where E wins the game if the TM accepts its input and U otherwise; this is well known to be equivalent to the more formal definition of acceptance in terms of the tree of all possible computations (Stockmeyer and Chandra, 1979). Thus we use "E wins" or "U loses" as synonyms of "the machine accepts its input" and conversely "U wins" or "E loses" as synonyms of "the machine does not accept" and we can rephrase the definition

of acceptance in "dead end" positions by saying that a player unable to make a legal move loses. This point of view shows clearly how reductions from ATMs can be used to derive results about "real" games such as chess, checkers, and Go.

2. Results

2.1. *Time Complexity*

THEOREM 1. *RLATM t time = NLATM t time = ATM t time.*

Proof. The fact that ATM t time is included in the other two classes is immediate. The opposite inclusion is proved by describing a linear time ATM simulation of $R(N)$LATM computations. The fact that such a simulation can be carried out in time $O(t^2)$ by copying to an archive tape all the configurations of the $R(N)$LATM and deterministically checking each step for compliance with the looping restriction is fairly obvious; to make the simulation run in time $O(t)$ we make the checking use the alternation mechanism.

A $R(N)$LATM with n tapes will be simulated by an ATM with $n+1$ tapes. The first n tapes contain exactly the same symbols as in the simulated machine while the $(n+1)$th "archive" tape is used to record for each step of the simulated machine a description of the action performed, in the format old-state, new-state, (symbol-read-on-tape-i, symbol-written-on-tape-i, direction-of-head-movement-on-tape-i) $(i=1\cdots n)$. If a player violates the looping restriction, these descriptions can easily be read back, performing a reversed computation which can be used to check for the violation. We describe here what happens after a universal step; for the equivalent process after an existential step interchange E and U. Player E has the option of initiating this reversing process (after making a copy of the current configuration); at any stage in the reverse computation, E may cause a comparison of the current and the copied configurations and wins if they are the same; U wins (in the RLATM case) if both clearing states are passed through in the reverse computation or (in either case) if it returns to the starting configuration. Thus E loses, if he initiates the reversing process without a violation having occurred.

2.2. *Upper Bounds on R(N)LATM S Space*

Note that the titles of this section and the following one need to be read carefully. This section refers to upper bounds on the set of problems solvable in $R(N)$LATM S space and therefore to lower bounds on the space required for $R(N)$LATMs to solve a problem of known deterministic complexity.

THEOREM 2. *NLATM S space* $\subseteq \bigcup_c$ *ATM* c^S *time.*

Proof. Since the NLATM has only c^S possible configurations, the ATM simulation described in the proof of Theorem 1 will run in time $O(c^S)$.

THEOREM 3. *RLATM S space* $\subseteq \bigcup_c$ *TM* c^{cS} *time.*

Proof. The restricted looping game being played by E and U can be regarded as a meta game without a looping restriction but with enormously more complex meta configurations. These meta configurations specify which of the clearing states was entered most recently and classify all configurations of the original game according to whether they have occurred since the last entry into none, one, or both of the clearing states; given also the current configuration of the original game as part of the meta configuration, all the rules of the original game including the looping restriction are rules of the meta game of the standard type which simply regulate which meta configurations can be transformed into which by legal meta moves. The number of meta configurations is $O(3^{c^S})$ and standard techniques are known for determining the outcome of such games in time $O(\text{number of configurations}^2)$ on a unit cost RAM whose registers all contain values bounded by a polynomial in the number of configurations (Fraenkel, 1980) which can then be simulated by a Turing machine with at worst a polynomial increase in the time (Aho, Hopcroft, and Ullman, 1974).

The results of Section 2.3 will show that the upper bounds on $R(N)$LATM S space proved in Theorems 2 and 3 cannot be tightened by more than a constant factor multiplying S.

2.3. *Lower Bounds on R(N)LATM S Space*

In this section we establish lower bounds on RLATM S space and on NLATM S space by showing how a RLATM (NLATM) running in space S can simulate an ATM computation using space (time) exponential in S. The two proofs are very simular so we will first state both theorems and then give a single proof for the RLATM case with notes on the differences in the NLATM case.

The assumption on S needed to prove the theorems depends on whether the space complexity is defined to include the space used on the read only input tape; if it does then $S(n) = \Omega(n)$ is needed and otherwise only $S(n) = \Omega(\log n)$.

THEOREM 4. *NLATM S space* $\supseteq \bigcup_c$ *ATM* c^S *time.*

THEOREM 5. *RLATM S space* $\supseteq \bigcup_c$ *ATM* c^S *space.*

2.3.1. *Overview of the proof.* A language recognised by an ATM in space (time) $O(c^S)$ is recognised also by an ATM subject to a similar space (time) bound, possibly for a different c, and also to two further restrictions:

(i) it has only one tape,

(ii) its head movements consist of a number of sweeps alternately left to right and the reverse, each of which covers, the whole of the tape used so far (perhaps extending the used section of tape further right) and its direction of movement at each step is uniquely determined by the current state and symbol read.

We will describe a simulation of such an ATM by a two tape $R(N)$LATM with space bound $O(S)$. The simulation has some similarity to the ATM simulation of the $R(N)$LATM described in the proof of Theorem 1 but with two major differences: first this simulation does not keep any explicit record of the configuration of the simulated machine but only keeps a record of steps like the archive tape of Theorem 1; second, instead of concatenating records of steps to the archive tape, each record overwrites the record of the previous step. Should either player write a move record which is inconsistent with earlier steps of the ATM, the looping restriction can be used to detect and punish the error.

The simulation proceeds in three phases, namely initialisation, simulation proper, and (optionally) challenging. The simulation proper consists of a sequence of TM steps corresponding to each step of the simulated ATM. This sequence consists of, first, the relevant player (e.g., U for a universal ATM step) writing on tape 1 a purported record of the ATM step and, second, an opportunity for the opposing player to challenge the validity of the described step, entering the challenging phase; if a step record is written of a step in which the ATM allegedly halts and this record is not challenged, then the simulating machine also halts accepting or not according to the final state of the ATM. Thus the simulating machine accepts exactly the same inputs as the simulated machine, provided the challenging process ensures that a player challenging a step record wins if and only if the described step was invalid. Tape 2 of the simulating machine contains the input and is never altered.

The format of tape 1 containing a record of one ATM step and some global information about the computation is illustrated in Fig. 1. First four single symbol fields to be referred to as old-state, symbol-read, new-state, and symbol-written will hold this information about the ATM step.

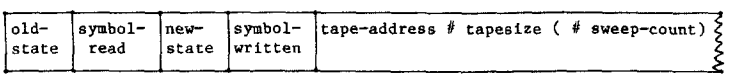

FIG. 1. Layout of tape 1.

Then are two (three for the NLATM) variable-sized fields separated by # symbols; they hold, respectively, tape-address the address on the tape at which this step took place, tape-size the length of the section of tape used up to this step (and in the NLATM case sweep-count the number of sweeps of the tape completed so far). The space for the variable sized fields is $O(S)$. (Because in the RLATM case it is $O(\log(\text{space used by ATM}))$ and in the NLATM case $O(\log(\text{number of sweeps}))$.)

Most of the simulation is carried out by states for which there is only one possible action for each pair of symbols read on the two tapes; these states will be referred to as deterministic although they must actually be classified as existential or universal. In particular the clearing states of the RLATM are of this type so that the argument does not depend on whether they are universal, existential, or a combination of the two.

The simulation of a single ATM step proceeds as follows starting with the TM read/write head deterministically placed over the symbol-read field:

(i) In either existential or universal states (according to the current state of the ATM) symbols are written in the symbol-read, new-state, and symbol-written fields.

(ii) In deterministic states, the four fixed-size fields are checked for consistency with the ATM and, if they are not consistent, the player who made the decisions loses.

(iii) The opposite player (e.g., U if the step was existential) has the opportunity to initiate the challenging process and otherwise returns to a deterministic state.

(iv) The address field is updated and special actions carried out if it is zero (at the left-hand end of the ATM tape) or greater than or equal to the tape-size field; if it is zero, one of the two clearing states is entered and promptly left again (or in the NLATM case the sweep-count is incremented); if it equals the tape-size, the other clearing state is entered and left (or the sweep-count incremented); if it is greater than the tape-size, the tape-size is incremented, the second clearing state is again entered (not for NLATM) and the appropriate symbol is placed in the symbol-read field (this symbol being the blank symbol unless the input tape has not yet been read to exhaustion in which case the next symbol is copied from it). (If step (iv) causes one of the variable sized fields to overflow into a # separator, the remaining fields will need to be shifted right to leave space.)

(v) The new-state field is copied into the old-state field and the read/write head is placed over the symbol-read field (or the new-state field in case the symbol-read has just been determined in step (iv)) and the appropriate state entered to start simulating the next ATM step.

It should be clear that, given the restrictions on the simulated ATM, this

process can be carried out with the deterministic states having access to enough information to do what they need. Thus the simulation will correctly simulate the ATM computation until E or U either writes in the symbol-read field a symbol other than the one last written in the appropriate cell or initiates the challenging process; the form of the challenging process will ensure that the first player to do either of these will lose.

2.3.2. *Challenging false (and true) claims.* The idea of the challenging process is very simple: If a player writes in the symbol-read field a symbol which was not the last one written by the ATM in this tape cell, the opposing player starts a sequence of steps which eventually must return to the configuration which existed at stage (iii) of the simulation of the ATM step which wrote the symbol; since the looping restriction prevents this, the player about to move at that point loses and the sequence is designed so that this is the player who made the initial mistake. A little gadgetry is needed to ensure that a player making an unsubstantiated challenge loses. We will describe the form of the challenging process after a universal step of the simulated ATM. The form after an existential step is exactly similar with existential and universal states interchanged.

At step (iii) of the simulation of a universal step, the machine enters an existential state called E_1, with the read/write head of tape 1 at location 1. The normal course of the simulation is for E to change to the universal state U_2 and for U then to change to a deterministic state to start step (iv) but in E_1 and U_2 there are other options available as illustrated in Fig. 2. In E_1 there is a suicidal option, namely to change to U_3 in which U has no alternative but to change to E_4; from E_4 the only possible transition is to E_1 but that is ruled out by the looping restriction and so E loses.

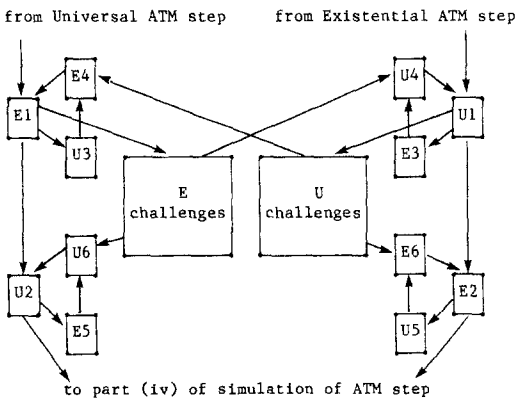

FIG. 2. Changes of state in the challenging process.

A similar suicidal option exists from U_2 via E_5 and U_6. The other option in E_1 is to challenge, that is to initiate a sequence of steps which can produce a tape configuration identical to what would have occurred during the simulation of a previous ATM step incompatible with this claimed one.

These challenging steps must write in the symbol-written field a symbol different from the current symbol-read field and then may change the tape-size, old-state, new-state, and symbol-read fields at will (not increasing tape-size), remove leading zeros from tape-address and finally move the read head of the input tape any distance leftwards; then they (in the NLATM case subtract one from the sweep-count removing a leading zero if one is created and) position the read/write head of tape 1 at position 1 and enter state U_4 or U_6. Thus E has had the opportunity to return the two tapes to exactly the configuration which would have existed at step (iii) of a previous ATM step which wrote a symbol at the current tape address different from the one which U claims was read there; E will now win iff he has done so and the step in question was the last one to write in this tape cell. For instance, if the previous step was an existential step, the machine will have entered state U_4 (see Fig. 2) and U loses because the transition to U_1 is prevented by the looping restriction; but, from U_4 without the configuration representing a previous step, U could enter U_1 and then E_3 causing E to lose.

The difference between the RLATM and NLATM cases lies in the mechanism which ensures that the only step relevant to the challenging process is the last one to have written in this tape cell. In the RLATM case, the fact that the two clearing states are entered, one at each end of a sweep over the tape, ensures that one clearing state has been entered since the last step at this address but both have been entered since the second last; in the NLATM case, the challenging process must reduce the sweep count by one producing a configuration which could only have occurred on the previous sweep.

3. Conclusions

Having established the lower bounds of Theorems 4 and 5 on $R(N)$LATM S space in terms of ATM complexity, we interpret them and Theorem 2 in the light of the results of Chandra *et al.* (1981) in deterministic terms and, taking into account also Theorem 3 which is already in deterministic terms, we finally conclude

$$\text{NLATM } S \text{ space} = \bigcup_c \text{TM } c^S \text{ space} = \text{TM } 2^{O(S)} \text{ space}$$

and

$$\text{RLATM } S \text{ space} = \bigcup_c \text{TM } c^{c^S} \text{ time} = \text{TM } 2^{2^{O(S)}} \text{ time}.$$

Theorems 2 and 4 can be used also to deduce the exponential space completeness of any game G of perfect information with a rule forbidding repetitions, whose configurations can be described in polynomial space, provided there exists an accurate reduction from ATM configurations to G configurations, that is a reduction such that

(1) To each ATM configuration C, there exists a unique G configuration $f(C)$, and $C \neq C'$ implies $f(C) \neq f(C')$.

(2) If, in the ATM configuration C, player $E(U)$ has lost, then in $f(C)$ player $E(U)$ is bound to lose against optimum play by his opponent.

(3) If, in the ATM configuration C, player $E(U)$ is to move and can move to C_i $(1 \leqslant \leqslant n)$, then in $f(C)$ player $E(U)$ is to move and can choose between n strategies which will result either in his opponent losing or in reaching $f(C_i)$ $(1 \leqslant i \leqslant n)$ via a fixed set of configurations known as PATH $(f(C), f(C_i))$. If the player to move does not choose one of these strategies, then he loses.

(4) $\mathrm{PATH}(A, B) \cap \mathrm{PATH}(C, D) = \varnothing$ unless $A = C$ or $B = D$. If $\mathrm{PATH}(A, B) \cap \mathrm{PATH}(C, B) \neq \varnothing$ then in traversing a path from A to B or from C to B, the first move into this intersection is made by the same player who would make the final move into B.

In fact such reductions would exist using the already published reductions to the games G_1, G_2, G_3 defined in (Stockmeyer and Chandra, 1979) and thus to chess (Fraenkel and Lichtenstein, 1981) and checkers (Robson, 1984a) except for one minor snag: the original reduction from G_1 to G_2 violates condition (1). However, this violation is easily removed (for each variable in the original set X, the reduction produces four variables $x_{i,1}$, $x_{i,2}$, $y_{i,1}$, and $y_{i,2}$; when player I changes $x_{i,1}$ or $x_{i,2}$, player II must change one of $y_{i,1}$ or $y_{i,2}$ but it is immaterial which he chooses; removing $y_{i,2}$ and requiring player II to change $y_{i,1}$ leaves a valid reduction; similar comments apply to the variables $x_{i,2}$ corresponding to original Y variables), establishing exponential space completeness for G_1, G_2, G_3, chess, and checkers with a no-looping rule.

The list of games whose non-looping version is known to be exponential space complete does not contain Go, whose non-looping version as played in China started this investigation. The reductions to Go from G_3 (Robson, 1983) make it easy for one player to force the other to break the non-looping rule.

Finally it is of interest to note that any NLATM computation can be regarded as an instance of the "generalised geography game" (Lichtenstein and Sipser, 1980), which is known to be complete in polynomial space. Thus Theorem 4 could be interpreted as implying that the geography game is still very hard (requiring space Ω (number of vertices)^c for $c > 0$) even

when restricted to apparently simple graphs which can be described very succinctly and even when this short description is provided as part of the input. Comparable results relating to other problems on graphs are presented by Wagner (1984).

RECEIVED: July 23, 1985; ACCEPTED: August 12, 1985

REFERENCES

AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. (1974), "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass.

CHANDRA, A. K., KOZEN, D. A., AND STOCKMEYER, L. J. (1981), Alternation, *J. Assoc. Comput. Mach.* **28**, 114–133.

FRAENKEL, A. S. (1980), From Nim to Go, *Ann. Discrete Math.* **6**, 137–156.

FRAENKEL, A. S., AND LICHTENSTEIN, D. (1981), Computing a perfect strategy for $n \times n$ chess requires time exponential in n, *J. Combin. Theory Ser. A* **31**, 199–213.

LICHTENSTEIN, D., AND SIPSER, M. (1980), Go is polynomial-space hard, *J. Assoc. Comput. Mach.* **27**, 393–401.

ROBSON, J. M. (1983), The complexity of go, *in* "Information Processing '83" (R. E. A. Mason, ed.), pp. 413–418, Internat. Fed. Inform. Process. Elsevier, New York.

ROBSON, J. M. (1984a), N by N checkers is exptime complete, *SIAM J. Comput.* **13**, 252–267.

ROBSON, J. M. (1984b), Combinatorial games with exponential space complete decision problems, *in* "Proceedings of 11th Symposium on Mathematical Foundations of Computer Science," pp. 498–506, Springer-Verlag, Berlin/New York.

STOCKMEYER, L. J., AND CHANDRA, A. K. (1979), Provably difficult combinatorial games, *SIAM J. Comput.* **8**, 151–174.

WAGNER, K. (1984), The complexity of problems concerning graphs with regularities, *in* "Proceedings of 11th Symposium on Mathematical Foundations of Computer Science," pp. 544–552, Springer-Verlag, Berlin/New York.